



PATENT ABSTRACTS OF JAPAN

(11) Publication number: **2000112742 A**(43) Date of publication of application: **21.04.00**

(51) Int. Cl.

G06F 9/06(21) Application number: **10285139**(22) Date of filing: **07.10.98**(71) Applicant: **FUJITSU LTD**

(72) Inventor:
KONDO TATSUO
UEHARA TADAHIRO
NAKAYAMA HIROKO
YAMAMOTO RIEKO
OHASHI KYOKO
YOSHIDA HIROYUKI

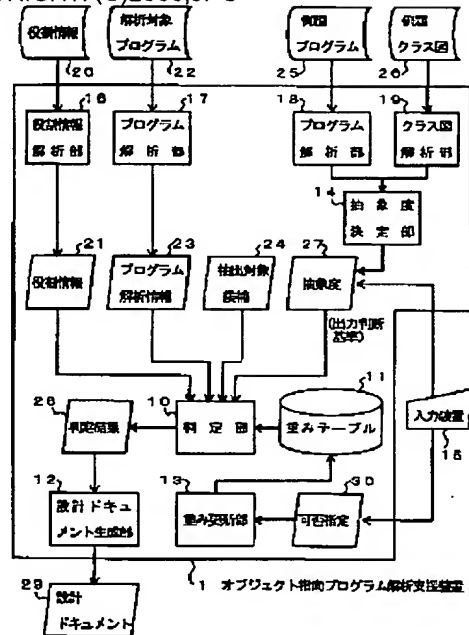
(54) **OBJECT-ORIENTED PROGRAM ANALYSIS
 SUPPORTING DEVICE AND ITS PROGRAM
 RECORDING MEDIUM**

(57) Abstract:

PROBLEM TO BE SOLVED: To improve the precision of judgement by automatically judge whether to extract information which should be reflected to a designing document in an object oriented program analysis supporting device analyzing an object- oriented program to support generation of the designing document.

SOLUTION: A judging part 10 evaluates the importance of the class/member candidate of an extracting object from program analysis information 23 obtained from a program analyzing part 18 and role information 21 obtained from a role information analyzing part 16 to select a candidate to extract according to inputted abstraction degree 27. A designing document generation part 12 generates the designing document 29 such as a class drawing based on a result selected by this part 10 and outputs it. Thus, the document 29 of an abstraction degree according to a purpose is generated.

COPYRIGHT: (C)2000,JPO



THIS PAGE BLANK (USPTO)

JP-A-2000-112742

**Object-oriented Program Analysis Supporting Device and
Program Storage Medium thereof**

5 [0064]

Next, the process flow of each unit of the present invention is described with reference to Figs. 14 through 17. Fig. 14 is a flowchart showing the process of a determination unit 10. Firstly, the determination
10 unit 10 obtains extraction candidates 24 (step S1). Then, the unit 10 checks whether a criterion (degree of abstract) is given from the outside (step S2). If the criterion is not given from the outside, the unit 10 uses a standard criterion (step S3). If the criterion
15 is given from the outside, the unit 10 obtains the criterion (step S4). Furthermore, the unit 10 checks whether there is a weight table 11 (step S3). If there is not the weight table 11, the unit 10 sets all weights (weighted values) equal (step S6). If there is the weight
20 table 11, the unit 10 obtains the weight table 11 (step S7).

[0065]

Then, the unit 10 obtains all criteria (step S8), and extracts one of the extraction candidates 24 (step
25 S9). Then, the unit 10 extracts one criterion of degree

THIS PAGE BLANK (USPTO)

of importance (step S10), computes the respective degree of importance of all the candidates and totals the respective degree of importance with weights (step S11). Furthermore, the unit 10 checks whether each candidate
5 has role information 21 (step S12). Only when a candidate has role information 21, the unit 10 computes the degree of importance of the role information 21 and totals the respective degree of importance of the role information of all the candidates (step S13). The processes in steps
10 10 through 13 are repeated until the computation of all the criteria are completed (step S14), and the unit 10 determines whether each extraction candidate passes or fails by comparing its computed result with the criteria. In other words, the unit 10 determines whether each of
15 the currently focused extraction candidates 24 should be outputted (step S15). The unit 10 repeats the processes in steps S9 through S15 until the processes of all the extracted candidates 24 are completed (step S16), and outputs a determined result (step S17).
20 [0066]

Fig. 15 is a flowchart showing the respective processes of program analyzing unit 17 and 18. The program analyzing unit 17 extracts classes from an analysis target program 22 (step S21), and extracts the
25 inheritance relationship between classes for each class

THIS PAGE BLANK (USPTO)

(step S22). Furthermore, the unit 10 extracts members for each class (step S23), and extracts both visibility and virtuality for all the members (step S24).

[0067]

5 The program analyzing unit 18 also performs the same process for an example program 25. These program analyzing processes are prior arts. Fig. 16 is a flowchart showing the process of a role information analyzing unit 16.

10 [0068]

 The role information analyzing unit 16 extracts design pattern names from role information 20 (step S31), extracts classes constituting a pattern (step S32) and extracts both the general and inheritance relationships
15 between classes (step S33). Then, the unit 16 extracts members which belong to each class (step S34) and extracts the meanings of each class and member (step S35). Then, the unit 16 inputs the result to the determining unit 10 as role information 21.

20 [0069]

 Fig. 17 is a flowchart showing the process of a class diagram analyzing unit 19. The class diagram analyzing unit 19 extracts classes from an example class diagram 26 (step S41) and extracts the inheritance
25 relationship between classes for each class (step S42).

THIS PAGE BLANK (USPTO)

Then, the unit 19 extracts members for each class (step S43), and extracts both visibility and virtuality for all the members (step S44). In this class diagram analyzing process, a prior art can be used.

5 [0070]

By collating the result analyzed by the program analyzing unit 18 with the result analyzed by the class diagram analyzing unit 19, it can be determined that what degree of abstract the structure of the example
10 program 25 is expressed in the example class diagram 26. An abstract degree determining unit 14 performs such a process. Since the process of generating a design document, such as a class diagram and the like from a real object-oriented program is already known, its
15 detailed description is omitted here. The major difference between the process of the design document generating unit 12 of the present invention and the conventional design document generating process is that in the conventional process, its target output is only
20 full information obtained from a program or specific predetermined information, while in the present invention, information to be outputted is restricted based on a determined result 28 according to a given degree of abstract 27, and a design document 29 is
25 generated in a necessary concept level.

THIS PAGE BLANK (USPTO)

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-112742

(P2000-112742A)

(43) 公開日 平成12年4月21日 (2000.4.21)

(51) Int.Cl. ⁷	識別記号	F I	テーマコード* (参考)
G 0 6 F 9/06	5 3 0	G 0 6 F 9/06	5 3 0 U 5 B 0 7 6
	5 4 0		5 3 0 P
			5 4 0 U

審査請求 未請求 請求項の数 5 O L (全 12 頁)

(21) 出願番号 特願平10-285139

(22) 出願日 平成10年10月7日 (1998.10.7)

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番
1号

(72) 発明者 近藤 竜生

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(72) 発明者 上原 忠弘

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(74) 代理人 100087848

弁理士 小笠原 吉義 (外2名)

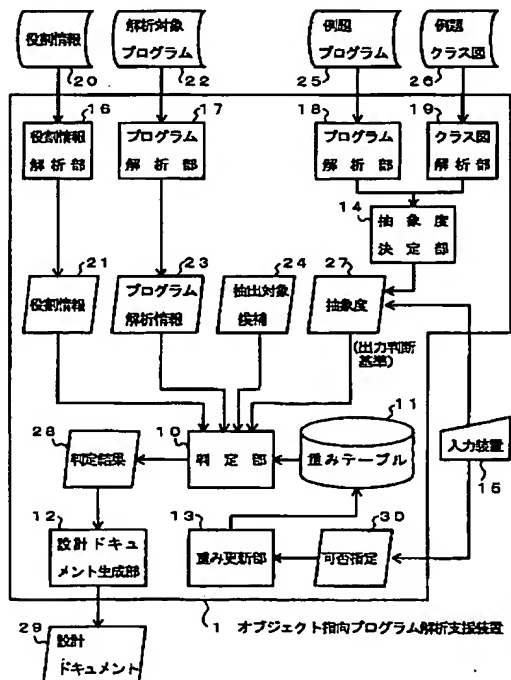
最終頁に続く

(54) 【発明の名称】 オブジェクト指向プログラム解析支援装置およびそのプログラム記録媒体

(57) 【要約】

【課題】 オブジェクト指向プログラムを解析して設計ドキュメントの生成を支援するオブジェクト指向プログラム解析支援装置に関し、設計ドキュメントに反映すべき情報を抽出するか否かの判断を自動的に行い、判断の精度を向上させる。

【解決手段】 判定部10は、プログラム解析部27から得たプログラム解析情報23または役割情報解析部16から得た役割情報21から、抽出対象のクラス/メンバ候補の重要度を評価し、入力した抽象度27に応じて抽出する候補を選択する。設計ドキュメント生成部12は、この判定部10が選択した結果に基づいてクラス図等の設計ドキュメント29を生成し出力する。これにより、目的に応じた抽象度の設計ドキュメント29が生成される。



【特許請求の範囲】

【請求項 1】 オブジェクト指向プログラムを解析して設計ドキュメントを生成するオブジェクト指向プログラム解析支援装置であって、オブジェクト指向プログラムを構成するクラスおよびメンバの抽出対象候補と、生成すべき設計ドキュメントの抽象度とを入力とし、前記各抽出対象候補の重要度と前記抽象度との比較により、前記各抽出対象候補を設計ドキュメントに記述すべきかを判断する判定手段を持つことを特徴とするオブジェクト指向プログラム解析支援装置。

【請求項 2】 請求項 1 に記載するオブジェクト指向プログラム解析支援装置において、前記判定手段は、プログラム解析情報、またはクラスもしくはメンバの役割情報から得られる複数の重要度の各々に重みを付け、それにより前記各抽出対象候補の総合的な重要度を決定することを特徴とするオブジェクト指向プログラム解析支援装置。

【請求項 3】 請求項 2 に記載するオブジェクト指向プログラム解析支援装置において、前記判定手段の結果に対する利用者からの可否の入力を得て、その可否の結果により、前記重要度に対する重み付けを学習する手段を持つことを特徴とするオブジェクト指向プログラム解析支援装置。

【請求項 4】 請求項 1 に記載するオブジェクト指向プログラム解析支援装置において、前記判定手段が用いる抽象度を、例題となるオブジェクト指向プログラムおよびそのクラス図から推測する抽象度決定手段を持つことを特徴とするオブジェクト指向プログラム解析支援装置。

【請求項 5】 オブジェクト指向プログラムを解析して設計ドキュメントの生成を支援する装置を計算機によって実現するためのプログラムを記録した記録媒体であって、オブジェクト指向プログラムを構成するクラスおよびメンバの抽出対象候補と、生成すべき設計ドキュメントの抽象度とを入力とし、前記各抽出対象候補の重要度と前記抽象度との比較により、前記各抽出対象候補を設計ドキュメントに記述すべきかを判断する処理を、計算機に実行させるプログラムを記録したことを特徴とするオブジェクト指向プログラム解析支援プログラム記録媒体。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】 本発明は、オブジェクト指向プログラムの解析をコンピュータを利用して支援するオブジェクト指向プログラム解析支援装置およびそのプログラム記録媒体に関する。

【0002】

【従来の技術】 オブジェクト指向プログラムの開発はインクリメンタルに行われる。すなわち、最初にプログラムの核となる部分のみを分析・設計し、それに基づいて

実装を行い（第 1 フェーズ）、プログラムの核となる部分の動作確認の後、プログラム拡張部分の分析・設計（第 2 フェーズ）へと進めていく。

【0003】 このとき、第 1 フェーズの実装工程において、当初の設計にはなかった変更が行われることがある。このような場合には、第 2 フェーズの開始にあたって、第 1 フェーズの設計に実装時の変更を反映したものを第 2 フェーズ設計の基礎とする必要がある。そうでなければ、設計と実装とが食い違ってしまう、実際に作成されたプログラムは設計と異なってしまうという問題が発生する。

【0004】 また、プログラムは、実装される機種や OS 等にかかわるような、クラス図等の設計ドキュメントよりもはるかに大量の情報を含む。したがって、第 1 フェーズのプログラム作成時点での変更を第 2 フェーズの設計ドキュメントに反映させる際に、プログラムから抽出した情報が、設計ドキュメントに反映されるべき本質的なものであるか、いわば実装詳細のような設計ドキュメントには反映すべきでないものであるかを判断する必要がある。

【0005】 従来、オブジェクト指向プログラムの開発を支援するために、クラス図やメッセージのシーケンス図からプログラムを可能な範囲で自動生成する技術が知られている。一方、これとは逆に、オブジェクト指向プログラムの解析を容易にするために、既存のオブジェクト指向プログラムからクラスやクラス相関等の情報を抽出し、UML (Unified Modeling Language) によるクラス図、シーケンス図等の設定ドキュメントを生成し、出力する方法も考えられている。

【0006】

【発明が解決しようとする課題】 しかし、この従来のオブジェクト指向プログラムの解析を支援する方法の場合には、プログラムから実装詳細情報を含むすべての情報が抽出されてしまうため、結局、設計ドキュメントに必要な情報であるかどうかの判断は、人間が行わなければならない。その判断には、設計やプログラミングについて高度な理解を要するという問題があった。また、大規模かつ複雑なシステムの開発にあっては、正確な判断を保証することがきわめて困難であるという問題もあった。

【0007】 本発明は、上記問題点の解決を図り、従来は人手で行われていたプログラム変更点の判断を自動化することによって、プログラムの変更点の設計ドキュメントへの反映を容易にし、判断の精度を向上させることを目的とする。

【0008】

【課題を解決するための手段】 本発明は、オブジェクト指向プログラムの開発環境においてプログラムを解析して設計ドキュメントを生成する装置において、オブジェクト指向プログラムを構成するクラスおよびメンバを入力とし、設計ドキュメントで記述すべきかを判断す

る手段を持つことを主要な特徴とする。判断基準として、生成するクラス図の抽象度を用いる。ここで、抽象度とは、どの程度まで詳しい情報を出力するか出力レベルを表すものである。

【0009】出力に対する抽象度が与えられると、プログラムを解析することにより獲得した抽出対象の重要度に応じて、設計ドキュメントへの出力要素が決定される。この重要度は、プログラム内外で与えられるクラスないしメンバの役割情報からも計算される。複数の重要度の各々に重みを付け、それにより総合的な重要度を決定する。

【0010】抽象度は、インタラクティブに指定されるが、既に生成済みのクラス図と、その基となったプログラムとを解析して、自動的に推測させることもできる。すなわち、例題クラス図と例題プログラムとから、それに合うような出力レベルでの抽象度が選択され、その抽象度に基づく出力が行われる。

【0011】さらに、一旦決定された判断に対して、利用者に可否を指定させ、それを学習して以後の判断に役立てるようにするための手段を持つ。

【0012】

【発明の実施の形態】図1は、本発明の実施の形態の構成例を示すブロック図である。オブジェクト指向プログラム解析支援装置1の判定部10は、解析対象のオブジェクト指向プログラムを構成するクラスまたはメンバ等の抽出対象候補24と、出力する設計ドキュメント29の出力判断基準である抽象度27とを入力して、解析対象プログラム22のプログラム解析情報23から計算した重要度と、プログラムのコメントその他から取得する役割情報21から計算した重要度とをもとに、その抽出対象候補24を抽出するかどうかを判断し、その判定結果28を出力する手段である。判定部10の入力として、永続的記憶に蓄えられている重みテーブル11が用いられる。重みテーブル11は、複数の重要度から総合的な重要度を定めるための各重要度に対する重み付けの値を保持する。

【0013】設計ドキュメント生成部12は、判定部10による判定結果28に基づいて、必要な抽象度27で設計ドキュメント29を生成し出力する。重み更新部13は、入力装置15を介して、出力された設計ドキュメント29の結果に対する利用者の可否指定30の情報を入力して、利用者が満足できる抽象度による設計ドキュメント29の出力がなされるように、必要に応じて重みテーブル11に記憶する重み付けを更新する。この重み付けの更新によって、重みが適当な値になるように学習される。重みテーブル11が更新されると、判定部10は、再度、抽出対象候補24の重要度を計算して、抽出対象候補24を選別する処理を行い、その結果を判定結果28として出力する。

【0014】抽象度27は、入力装置15からインタラ

クティブに入力することもできるが、例題プログラム25および例題クラス図26から推測して決定することもできる。抽象度決定部14は、入力装置15から抽象度を取得する代わりに、例題プログラム25および例題クラス図26を、それぞれプログラム解析部18とクラス図解析部19によって解析し、その解析情報から抽象度27を推測する手段である。

【0015】役割情報解析部16は、別ファイル形式の役割情報20または解析対象プログラム中のコメント等から抽出した役割情報を解析して、各クラスおよびメンバの意味などの役割情報21を判定部10に渡す。プログラム解析部17は、解析対象プログラム22を解析して、解析結果のプログラム解析情報23を判定部10に渡す。プログラム解析部18は、例題プログラム25を解析し、クラス図解析部19は、例題クラス図26を解析し、それぞれの解析結果を抽象度決定部14に渡す。プログラム解析部17、18、クラス図解析部19は、それぞれ電子文書の形式である解析対象プログラム22、例題プログラム25、例題クラス図26を解析するが、これらの手段は、公知技術により実現することができる。

【0016】図1に示す各処理手段を計算機によって実現するためのプログラムは、計算機が読み取り可能な可搬媒体メモリ、半導体メモリ、ハードディスクなどの適当な記録媒体に格納することができる。

【0017】以下、図2に示すようなオブジェクト指向プログラムを解析対象とする場合の処理について、本発明の実施の形態を詳しく説明する。なお、以下の説明において、クラス名、データメンバ名、メンバ関数名の表記は次のようにしている。

【0018】・クラス名

先頭のみ大文字の単語を直接結合したもの（例：Directory, AbstractFile）。

【0019】・データメンバ名

小文字のみからなる単語をアンダースコア（`_`）によって結合し、最後に括弧を伴わないもの（例：name, pipe_name）。

【0020】・メンバ関数名

小文字のみからなる単語をアンダースコア（`_`）によって結合し、最後に括弧を伴うもの（例：open(), get_name()）。

【0021】〔1〕抽出対象候補

図3は、抽出対象候補およびプログラム解析情報の例を示す図である。抽出対象候補は、オブジェクト指向プログラムを構成するクラスまたはメンバである。図3

(A)に示すように、図2の解析対象プログラムについてのクラスの候補は、AbstractFile, File, Directory, Socket, Pipeであり、図3(B)に示すように、メンバの候補は、name, permission, open(), close

`e()`, `create()`, `delete()`, `get_name()`, `get_permission()`, `get_all_files()`, `bind_port()`, `pipe_name`である。

【0022】これらの抽出対象候補は、解析対象プログラムからプログラム解析部17によってプログラム解析情報の一部として取得してもよく、また、別の入力手段により直接取得してもよい。

【0023】〔2〕プログラム解析情報

① クラスのプログラム解析情報

クラスのプログラム解析情報として、図3(A)に示すように、例えば、各クラスのメンバ数、他クラスとの関連数、クラスの継承の深さ等を用いる。

【0024】メンバ数として、メンバ数が多いクラスは構造が複雑であり、より重要であると考えられることから、クラスにはメンバ数に応じた重要度を与える。また、他クラスとの関連数では、他のクラスと密接な関係を持つクラスは重要であると考えられることから、例えば、他のクラスを型として持つデータメンバ数が多いクラスに高い重要度を与える。

【0025】ただし、FacadeパターンやAbstract Factoryパターンに現れるクラスのように、他のクラスとの関連がきわめて疎であるにもかかわらず、設計段階で記述すべき重要なクラスもある。これらのクラスについては、後述する役割情報を用いて重要度を与える。

【0026】なお、継承関係の数については、後述するように別項とするため、ここで対象とするのは集約(aggregation)を含む関連(association)のみである。

【0027】また、クラスの継承の深さでは、継承の深いクラスは、一般的に、より多くのメンバを含み、実装に関する詳細情報の記述という側面を持つ傾向が強い。したがって、深い継承関係が存在する場合には、上位(基底クラス)に出現するクラスほど設計の本質に近いと考えられるため、上位に出現するクラスに対して高い重要度を与える。

【0028】② メンバのプログラム解析情報

メンバのプログラム解析情報として、各メンバの公開範囲(`public/private`の種別)、メンバの抽象性、メンバ関数の複雑さや大きさ等を用いる。図3(B)は、メンバのプログラム解析情報の例を示す。

【0029】メンバの公開範囲では、`public`なメンバ関数は、オブジェクトの振舞いのインタフェースを決定する。したがって、設計ドキュメントで仕様として記述されるべきことが多い。そのため、高い重要度を与える。一方、`private`なメンバ関数は、他のメンバ関数が巨大になった場合に分割して処理の見通しをよくするために実装時に加えられることが多いため、これらについては低い重要度を与える。データメンバは、特

殊な事情がない限り`private`であるので、この基準については考えない。

【0030】また、メンバの抽象性では、類似した機能を持つ複数のクラスでアクセスインタフェースを統一するため、抽象メンバ関数を持つクラスを設け、それを継承するという設計技法がしばしば用いられる。このような場合に、抽象メンバ関数を変更すると他のクラスや関数への影響が非常に大きい。したがって、抽象メンバ関数は、ほとんどの場合に設計ドキュメントに記述されるべき重要性を持つ。

【0031】メンバ関数の複雑さおよび大きさについて、コンストラクタやデストラクタの処理は、メンバの初期設定や確保したメモリの解放であり処理は単純であると考えられる。また、データメンバを参照したり、値を設定したりするだけの関数(アクセス関数)も、単純な処理しか行わない。これら以外の、他のオブジェクトに問い合わせを行うメンバ関数や自身が持つ複数のデータメンバを関連付けて参照ないし更新するメンバ関数の処理は、複雑であるため高い重要度を与える。

【0032】また、非効率的なプログラミングを行っていない限り、複雑度とメンバ関数の大きさは正の相関関係を持つ。すなわち、複雑度の代わりにメンバ関数の行数を用いることも可能である。図3(B)に示すように、本例では、メンバ関数の複雑度を表すものとして、メンバ関数の行数を用いている。なお、行数が少なくても他のオブジェクトに対するアクセス回数が多いものは、処理が複雑であると考えられるので、このようなメンバ関数にも高い重要度を与える。

【0033】以上はメンバ関数の場合であり、データメンバの場合には、更新や参照の頻度、前後に参照される他のデータメンバの調査など、手続き解析処理を用いて重要度を決定する。処理の詳細はきわめて煩雑であるが、プログラムスライシング(program slicing)という既存の技術を用いて行うことができるので、ここでのこれ以上の詳しい説明は省略する。

【0034】〔3〕役割情報

本例では、役割情報としてデザインパターンの構造情報を用いる。デザインパターンとは、ガンマ(E. Gamma)らが著書「Design Patterns: Elements of Reusable Object-Oriented Software」でソフトウェア開発への適用を提示した、信頼性の高いシステムを効率的に開発するための概念であり、ソフトウェア設計において、高い頻度で現れる一群のクラスの集まり(パターン)を系統立てて収集し、これらのパターンを組み合わせることで設計の効率化を図るというものである。

【0035】デザインパターンを構成するクラスのうちいくつかは、パターン全体のインタフェースを決定するために重要であり、設計ドキュメントで記述されるべき性質のものである。しかし、設計時にはパターンが指定され、それがプログラムとして生成されるが、どのパ

ターンのどの部分としてそのクラスが設計されたのかという情報は、一般にはプログラムの構成要素としては残らない。

【0036】本例では、デザインパターンに基づく設計ツールにより、パターン名およびクラスのパターン内の位置付け（クラスの役割）をプログラム中のコメントないし別ファイルの形式で残すものとし、これを役割情報として利用する。

【0037】一例として、デザインパターンの中でも基本的なCompositeパターンについて説明する。図4は、Compositeパターンの例を示す図である。

【0038】Compositeパターンは、従来のプログラミング技法で言われるところの木構造を用いるときに現れるパターンであり、Client、Component、Leaf、Compositeの4個のクラスを持つ。

【0039】Clientクラスは、1つの木構造に対して、なんらかの要求を与えるものを示す。Componentクラスは、木構造の1つの節（葉もしくは部分木）を表し、これはComponentがLeafとCompositeを派生クラスとして持つということで表現される。Leafクラスは、木構造における葉を表し、節のうち、先に伸びる部分木を持たないものとして定義される。Compositeクラスは、先にはいくつかの節がついている部分木を表し、これはCompositeが任意個のComponentを持つということで表現される。

【0040】Compositeパターンにおいて、Component、Leaf、Compositeは、operation（開発者定義の任意の処理）、add（節の追加）、remove（節の削除）、get（節の検索）の4つの要求を処理できる。

【0041】図5は、役割情報の例を示す図である。図2に示す解析対象プログラムの中のクラスまたはメンバが、Compositeパターン内でどのクラス／メンバに相当するかをパターン内の位置付けとして抽出し、この位置付けにより重要度を与える。図5の例では、クラスについて、Componentクラスであれば重要度100、Compositeクラスであれば重要度80、Leafクラスであれば重要度70としている。また、メンバについて、Component.operation()であれば重要度100、Component.add()またはComponent.remove()であれば重要度70としている。

【0042】〔4〕判定部

④ クラスの重要度評価方法

図6は、クラスの重要度評価方法および重みテーブルの初期値の例を示す図である。

【0043】判定部10は、クラスの場合には、プログ

ラム解析情報のメンバ数、他クラスとの関連数、継承の深さの各項目の重要度、メンバ重要度および役割情報による重要度から、そのクラスの総合重要度を計算する。評価は以下のようにして行う。

【0044】メンバ数については、最もメンバの多いクラスを重要度100、少ないクラスを重要度0とし、メンバ数から最少を減算したものに比例した重要度を与える。他クラスとの関連数については、最も関連数の多いクラスを重要度100、少ないクラスを重要度0とし、関連数に比例した重要度を与える。継承の深さについては、最も継承の浅い（深さ0）クラスを重要度100、最も深いクラスを重要度0とし、深さを最大から減算したものに比例した重要度を与える。

【0045】メンバ重要度については、重要なメンバを持つクラスは重要であると考えられるため、そのクラスにおけるすべてのメンバについて重要度の平均をとる。役割情報による重要度については、クラスに対応する役割情報が与えられている場合は、その情報の持つ重要度をクラスに与える。

【0046】クラスの総合重要度として、上記の各項目に重みを掛けたもののうち最大のものを取る。ここでは、1つでも突出した重要度を持つクラスは、重要であるとして、その最大の重要度をそのクラスの総合重要度とする。

【0047】② メンバの重要度評価方法

図7は、メンバの重要度評価方法および重みテーブルの初期値の例を示す図である。

【0048】判定部10は、抽出対象候補がメンバの場合には、プログラム解析情報の公開範囲、メンバの抽象性、メンバの複雑さと大きさ（メンバ関数の行数）の各項目の重要度、役割情報による重要度から、そのメンバの総合重要度を計算する。評価は以下のようにして行う。

【0049】公開範囲については、メンバ関数が、publicならば重要度50、privateならば重要度10とする。抽象性については、抽象メンバ関数ならば重要度70、そうでなければ重要度20とする。メンバの複雑さと大きさとして、メンバ関数の行数を用い、メンバ関数の実行文が30以上ならば重要度100とし、30未満ならば実行文数に比例した重要度を与える。これはプログラムメトリクスの研究によると、良く設計されたプログラムにおいては、一つの関数の実行文の数は最大30前後であると言われていることによる。

【0050】役割情報による重要度として、メンバに対応する役割情報が与えられている場合は、その情報の持つ重要度をメンバに与える。メンバの総合重要度として、上記の各項目に重みを掛けたもののうち最大のものを取る。ここでは、1つでも突出した重要度を持つメンバは、重要であるとして、その最大の重要度をそのメンバの総合重要度とする。

【0051】図8は、以上のような評価方法により評価した結果の重要度の例を示す図である。図8(A)はクラスの重要度の例、図8(B)はメンバの重要度の例を示す。図6および図7に示すような評価方法に基づき、クラスのAbstractFileについては、メンバ数、関連数、継承の深さ、役割情報による重要度(役割重要度)の重要度が100であるので、総合重要度を100とする。また、Fileについては、役割重要度として重要度70が与えられているので、総合重要度を70とする。

【0052】判定部10は、以上のクラスの総合重要度およびメンバの重要度と、与えられた抽象度とから、設計ドキュメント29として出力すべき候補を抽出して、その判定結果28を設計ドキュメント生成部12に通知する。設計ドキュメント生成部12は、判定結果28に基づき、必要十分な詳しさを設計ドキュメント29を生成し出力する。

【0053】重み更新部13は、入力装置15を介して判定部10の処理結果に対する利用者の可否指定30の情報を入力する。“可”であれば設計ドキュメント生成部12が出力した設計ドキュメント29を最終出力とする。“不可”であれば、判定部10の出力に対して入力装置15から変更された内容を、新しい判定結果28として、設計ドキュメント生成部12へ出力する。この際に、入力装置15から加えられた変更内容を学習し、必要に応じて重みテーブル11に記憶する重み付けを修正する。

【0054】次に、代表的な設計ドキュメント29の出力例として、クラス図の出力例を説明する。図9～図11は、本実施の形態において出力されるクラス図の例を示す図である。

【0055】図9は、抽象度を20以下にした場合に出力されるクラス図の例である。図8の重要度の例に示すように、本例では候補の総合重要度の最低が20であるため、抽象度が20以下の出力が指定された場合には、すべてのクラスおよびメンバの抽出対象候補が選出されることになる。したがって、この場合に出力されるクラス図は、図2に示す解析対象プログラムのすべてのクラス構成を示す図になる。この出力結果は、従来技術による出力結果と同様になると考えてよい。

【0056】図10は、出力の抽象度を30以上にした場合に出力されるクラス図(実装設計レベル)の例を示す。この場合には、総合重要度が30以上の候補のみを抽出してクラス図が生成され、出力される。したがって、図9に示すクラス図では出力される重要度20のPipeクラスは、この場合には表現されない。

【0057】図11は、出力の抽象度を70以上にした場合に出力されるクラス図(分析設計レベル)の例を示す。この場合には、総合重要度が70以上の候補のみを抽出してクラス図が生成され、出力される。したがっ

て、図10に示すクラス図では出力されるSocketクラスは、重要度が70以下であるので、この場合には表現されない。

【0058】このように、利用者は、さまざまな抽象度を与えることにより、適切なレベルのクラス図を得ることが可能となる。

〔5〕抽象度決定部

抽象度は、入力装置15を介して利用者が指定した値を直接入力してもよく、また、例題プログラム25および例題クラス図26を入力として、プログラム解析部18およびクラス図解析部19により取得した解析情報から、抽象度決定部14により決定することもできる。抽象度決定部14は、具体的にはプログラム解析部18の解析結果について、判定部10と同様な重要度の計算を行い、どのレベルの重要度まで選択すれば例題クラス図26と同レベルのクラスまたはメンバが抽出されるかをチェックして、抽象度を推測する。

【0059】〔6〕重み更新部

判定部10および設計ドキュメント生成部12の処理により、クラス図等の設計ドキュメント29を生成して利用者に提示した後、重み更新部13は、入力装置15を介して利用者の可否指定30を、その理由とともに入力する。

【0060】重み更新部13は、入力した可否指定30の理由に基づいて重みテーブル11中の該当する項目を特定し、その重みの値を修正する。例えば、図11に示すクラス図を提示した後、出力結果に対して「publicなメンバ関数はなるべく抽出する」という否定理由が入力された場合には、メンバ関数がpublicである場合の重み付けの値を増加させる。

【0061】図12は、重みテーブルの変更の例を示す。この例では、「publicなメンバ関数はなるべく抽出する」という否定理由の入力により、メンバの公開範囲の重みを1から1.6に変更している。これにより、公開範囲のpublicに与えられる総合重要度を決定する際の重要度は、 $50 \times 1.6 = 80$ になる。

【0062】図13は、図11に示すクラス図について重み付けの値を修正した後に、出力されたクラス図の例を示している。すなわち、抽象度70以上の出力が指定された場合、publicメンバ関数の重み付けの値が1.6になると、その重要度が50から80になるので、重み付けの値の変更により、いままでは表現されていなかった、メンバのget_name()、get_permission()が表現される。

【0063】なお、複数の否定理由が与えられた場合の重み付け決定アルゴリズムは、例えばニューラルネットワーク技術として公知の技術を利用できるので、ここではその詳細な説明を省略する。

【0064】次に、図14～図17を参照して本発明の各手段の処理の流れを説明する。図14は、判定部10

の処理フローチャートである。判定部 10 は、まず抽出対象候補 24 を取得する（ステップ S1）。次に、外部から判断基準（抽象度）が与えられているかどうかを調べ（ステップ S2）、外部から判断基準が与えられていない場合には標準判断基準を使用し（ステップ S3）、外部から判断基準が与えられている場合にはその判断基準を取得する（ステップ S4）。さらに、重みテーブル 11 が存在するかどうかを調べ（ステップ S5）、重みテーブル 11 が存在しない場合には全ての重み（重み付けの値）を同一にし（ステップ S6）、重みテーブル 11 が存在する場合には重みテーブル 11 を取得する（ステップ S7）。

【0065】次に、全判定要素を取得し（ステップ S8）、抽出対象候補 24 を 1 つ取り出す（ステップ S9）。続いて、重要度の判定要素を 1 つ取り出し（ステップ S10）、判定要素についての候補の重要度を計算し、重み付きで合算する（ステップ S11）。さらに候補に対応する役割情報 21 があるかどうかを調べ（ステップ S12）、役割情報 21 がある場合にのみ役割情報 21 の重要度を計算して合算する（ステップ S13）。全ての判定要素について計算が終わるまで、ステップ S10～ステップ S13 の処理を繰り返し（ステップ S14）、計算結果を判断基準と比較して可否を決定する。すなわち、現在着目している抽出対象候補 24 を出力するか否かについて決定する（ステップ S15）。すべての抽出対象候補 24 について処理が終了するまで、ステップ S9～ステップ S15 の処理を行い（ステップ S16）、判定結果 28 を回答する（ステップ S17）。

【0066】図 15 は、プログラム解析部 17、18 の処理フローチャートである。プログラム解析部 17 は、解析対象プログラム 22 からクラスを抽出し（ステップ S21）、各クラスについてクラス間の継承関係を抽出し（ステップ S22）、さらに、各クラスについてメンバを抽出し（ステップ S23）、すべてのメンバについて可視性と仮想性を抽出する（ステップ S24）。

【0067】なお、プログラム解析部 18 も、例題プログラム 25 に対して同様の処理を行う。これらのプログラム解析処理は、既存の技術である。図 16 は、役割情報解析部 16 の処理フローチャートである。

【0068】役割情報解析部 16 は、役割情報 20 からデザインパターン名を抽出し（ステップ S31）、パターンを構成するクラスを抽出し（ステップ S32）、クラス間の関係および継承関係を抽出し（ステップ S33）、各クラスに属するメンバを抽出し（ステップ S34）、各クラスおよびメンバの意味を抽出する（ステップ S35）。その結果を役割情報 21 として、判定部 10 の入力とする。

【0069】図 17 は、クラス図解析部 19 の処理フローチャートである。クラス図解析部 19 は、例題クラス図 26 からクラスを抽出し（ステップ S41）、各クラ

スについてクラス間の継承関係を抽出し（ステップ S42）、各クラスについてメンバを抽出し（ステップ S43）、すべてのメンバについて可視性と仮想性を抽出する（ステップ S44）。このクラス図解析処理では、既存の技術を用いることができる。

【0070】プログラム解析部 18 による解析結果と、クラス図解析部 19 による解析結果とを照合すれば、例題クラス図 26 が、例題プログラム 25 の構造をどの程度の抽象度で表現しているかを判別することができる。抽象度決定部 14 は、その処理を行う。また、実際のオブジェクト指向プログラムからクラス図等の設計ドキュメントを生成する処理は、既知であるので、ここでの詳細な説明は省略する。本発明に係る設計ドキュメント生成部 12 の処理と、従来の設計ドキュメント生成処理との大きな違いは、従来、プログラムから得られるすべての情報またはあらかじめ定められた特定の情報だけを出力対象としていたのに対し、本発明では、与えられた抽象度 27 に応じた判定結果 28 に基づいて出力する情報を制限し、必要な概念レベルで設計ドキュメント 29 を生成する点である。

【0071】

【発明の効果】以上説明したように、本発明によれば、プログラム変更点を設計ドキュメントへ反映する場合に、利用者が所望する概念レベルに対応して、設計ドキュメントに反映すべき情報の抽出の判断を自動化することができる。これにより、設計ドキュメントへの反映作業の工数を低減でき、オブジェクト指向ソフトウェアの開発を効率化することが可能になる。特に、従来、設計ドキュメントへの反映では、設計およびプログラミングに対する高度な理解が必要であったのに対し、それが不要となり、複雑なシステムの開発において効果が大きい。

【図面の簡単な説明】

【図 1】本発明の実施の形態の構成例を示すブロック図である。

【図 2】解析対象プログラムの例を示す図である。

【図 3】抽出対象候補およびプログラム解析情報の例を示す図である。

【図 4】Composite パターンの例を示す図である。

【図 5】役割情報の例を示す図である。

【図 6】クラスの重要度評価方法および重みテーブルの初期値の例を示す図である。

【図 7】メンバの重要度評価方法および重みテーブルの初期値の例を示す図である。

【図 8】重要度の例を示す図である。

【図 9】出力されるクラス図の例を示す図である。

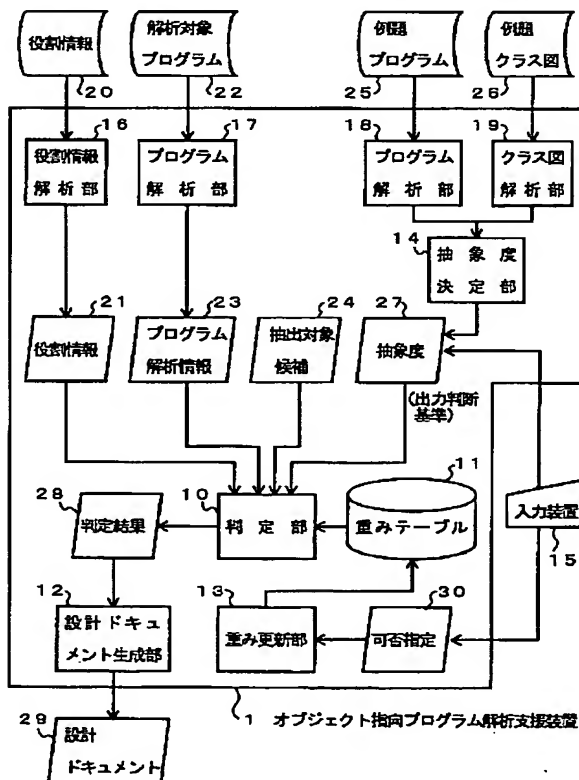
【図 10】出力されるクラス図の例を示す図である。

【図 11】出力されるクラス図の例を示す図である。

【図 12】重みテーブルの変更の例を示す図である。

- 1 5 入力装置
- 1 6 役割情報解析部
- 1 7 プログラム解析部
- 1 8 プログラム解析部
- 1 9 クラス図解析部
- 2 0 役割情報
- 2 1 役割情報
- 2 2 解析対象プログラム
- 2 3 プログラム解析情報
- 2 4 抽出対象候補
- 2 5 例題プログラム
- 2 6 例題クラス図
- 2 7 抽象度
- 2 8 判定結果
- 2 9 設計ドキュメント
- 3 0 可否指定

【图2】



解析対象プログラムの例

```
class AbstractFile {
private:
    String name;
    Permission permission;
public:
    virtual void open();
    virtual void close();
    virtual void create();
    virtual void delete();
    String get_name();
    Permission get_permission();
};

class File : public AbstractFile {
};

class Pipe : public File {
private:
    PipeName pipe_name;
};

class Socket : public File {
public:
    void bind_port();
};

class Directory : public AbstractFile {
private:
    vector<AbstractFile> files;
public:
    void get_all_files();
};
```

【図3】

抽出対象候補およびプログラム解析情報の例

(A) クラスの候補

名 前	メンバ数	他クラスとの関連数	継承の深さ
AbstractFile	8	1	0
File	0	0	1
Directory	1	1	1
Socket	1	0	2
Pipe	1	0	2

(B) メンバの候補

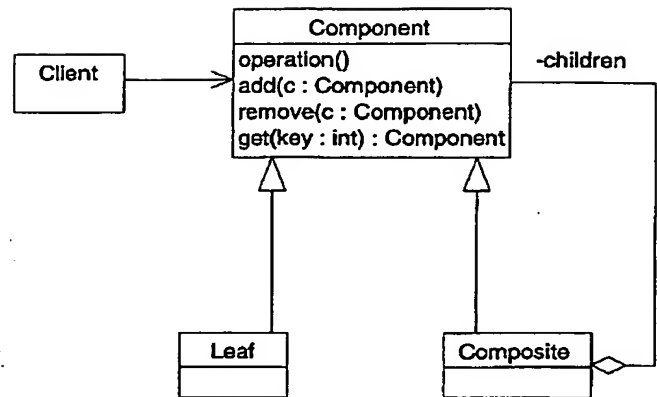
名 前	所属クラス	公開範囲	抽象性	メンバ定義の行数
name	AbstractFile	private	no	—
permission	AbstractFile	private	no	—
open()	AbstractFile	public	yes	5
close()	AbstractFile	public	yes	5
create()	AbstractFile	public	yes	8
delete()	AbstractFile	public	yes	8
get_name()	AbstractFile	public	no	1
get_permission()	AbstractFile	public	no	1
get_all_files()	Directory	public	no	3
bind_port()	Socket	private	no	10
pipe_name	Pipe	private	no	—

【図5】

役割情報の例

クラス名ないしメンバ名	パターン名	パターン内の位置付け	位置付けによる重要度
AbstractFile	Composite	Component	100
File	Composite	Leaf	70
Directory	Composite	Composite	80
AbstractFile.open()	Composite	Component.operation()	100
AbstractFile.close()	Composite	Component.operation()	100
AbstractFile.create()	Composite	Component.add()	70
AbstractFile.delete()	Composite	Component.remove()	70

【図4】



【図6】

クラスの重要度評価方法の例

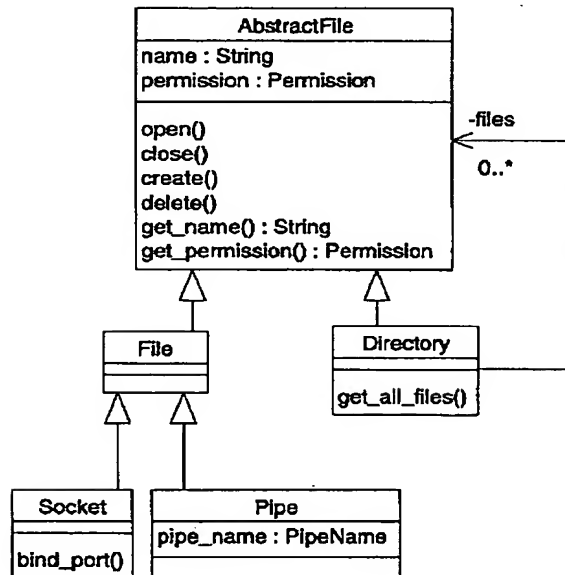
評価項目	重み初期値	評価方法
メンバ数	1	最もメンバの多いクラスを重要度100、少ないクラスを重要度0とし、メンバ数から最少を減算したものに比例した重要度を与える
他クラスとの関連数	1	最も関連数の多いクラスを重要度100、少ないクラスを重要度0とし、関連数に比例した重要度を与える
継承の深さ	1	最も継承の浅い(深さ0)クラスを重要度100、最も深いクラスを重要度0とし、深さを最大から減算したものに比例した重要度を与える
メンバ重要度	1	すべてのメンバについて重要度の平均をとる
役割情報による重要度	1	クラスに対応する役割情報が与えられている場合は、その情報の持つ重要度をクラスに与える
総合重要度	—	上記の各項目に重みを掛けたもののうち最大のものをとる。(1つでも突出した重要度を持つクラスは重要であるとする)

【図7】

メンバの重要度評価方法の例

評価項目	重み初期値	評価方法
公開範囲	1	publicならば重要度50、privateならば重要度10とする
抽象性	1	抽象メンバ関数ならば重要度70、そうでなければ重要度20とする
複雑さと大きさ	1	実行文が30以上ならば重要度100とし、30未満ならば実行文数に比例した重要度を与える
役割情報による重要度	1	メンバに対応する役割情報が与えられている場合は、その情報の持つ重要度をメンバに与える
総合重要度	—	上記の各項目に重みを掛けたもののうち最大のものをとる。(1つでも突出した重要度を持つメンバは重要であるとする)

【図9】



【図8】

重要度の例

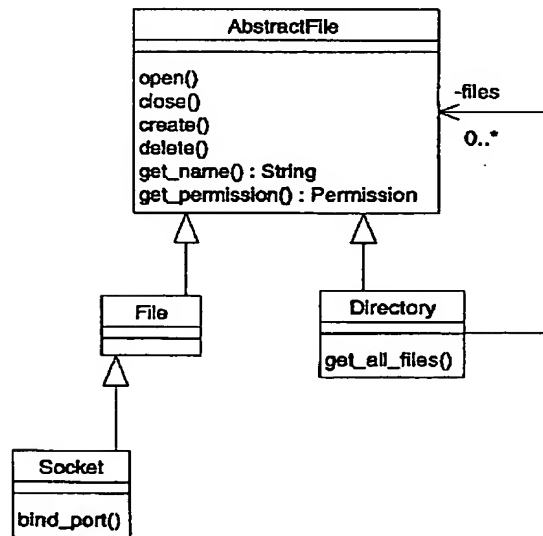
(A) クラスの重要度

名前/評価項目	メンバ数	関連数	継承の深さ	メンバ重要度	役割重要度	総合重要度
AbstractFile	100	100	100	60	100	100
File	0	0	50	0	70	70
Directory	12.5	100	50	50	80	100
Socket	12.5	0	0	33.3	0	33.3
Pipe	12.5	0	0	20	0	20

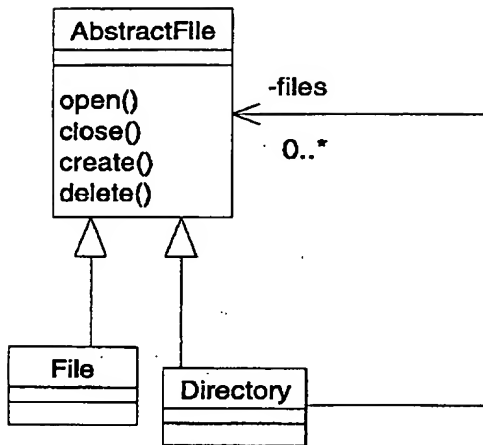
(B) メンバの重要度

名前/評価項目	公開範囲	抽象性	メンバ関数の行数	役割重要度	総合重要度
AbstractFile.name	10	20	—	0	20
AbstractFile.permission	10	20	—	0	20
AbstractFile.open()	50	70	16.7	100	100
AbstractFile.close()	50	70	16.7	100	100
AbstractFile.create()	50	70	26.7	70	70
AbstractFile.delete()	50	70	26.7	70	70
AbstractFile.get_name()	50	20	3.3	0	50
AbstractFile.get_permission()	50	20	3.3	0	50
Directory.get_all_files()	50	20	10.0	0	50
Socket.bind_port()	10	20	33.3	0	33.3
Pipe.pipe_name	10	20	—	0	20

【図10】



【図 11】

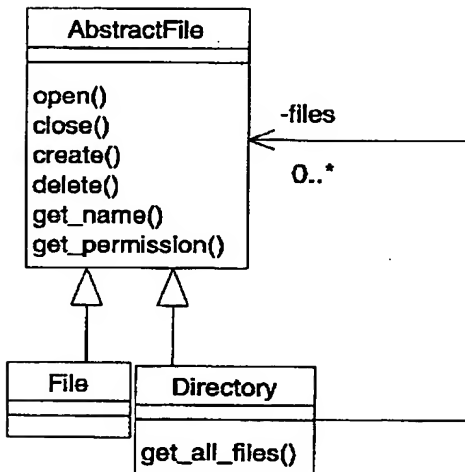


【図 12】

重みテーブルの変更の例

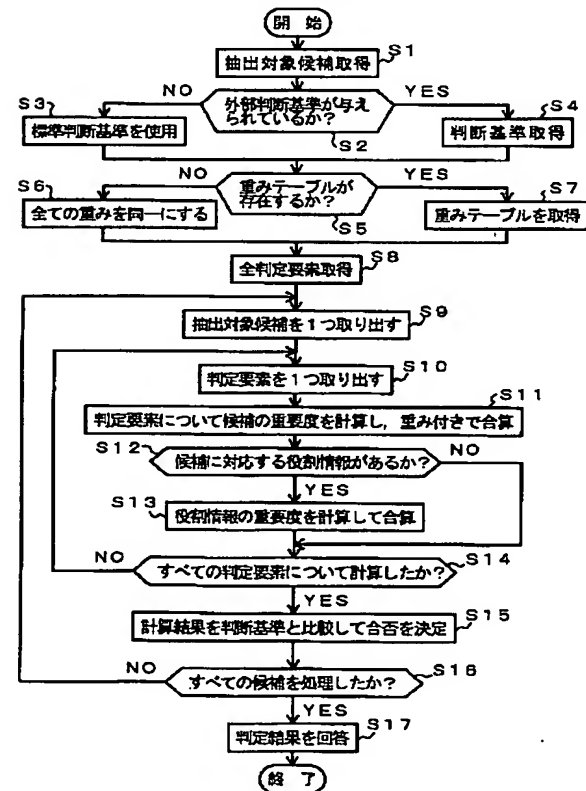
評価項目	重みの初期値	更新後の重み
クラス		
メンバ数	1	1
他クラスとの関連数	1	1
継承の深さ	1	1
メンバ重要度	1	1
役割情報による重要度	1	1
メンバ		
公開範囲	1	1.6
抽象性	1	1
複雑さと大きさ	1	1
役割情報による重要度	1	1

【図 13】



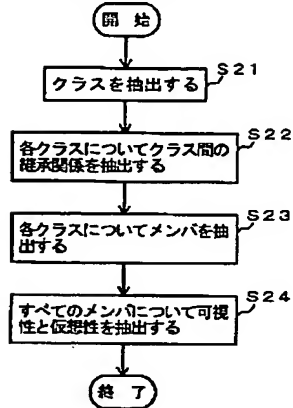
【図 14】

判定部の処理フローチャート



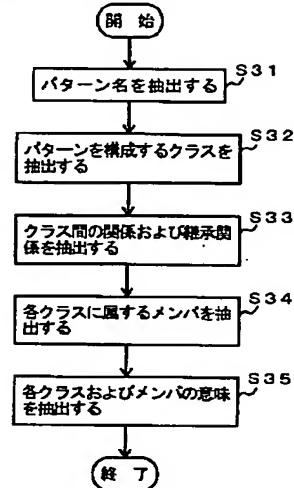
【図15】

プログラム解析部の処理フローチャート



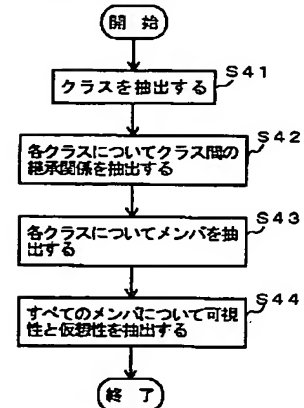
【図16】

役割情報解析部の処理フローチャート



【図17】

クラス図解析部の処理フローチャート



フロントページの続き

(72)発明者 中山 裕子

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(72)発明者 山本 里枝子

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(72)発明者 大橋 恭子

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(72)発明者 吉田 裕之

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

Fターム(参考) 5B076 DE04